Project Summary Report:

Bandwidth Estimations and Its Applications

Jin Guojun

Funding: 0.2 FTEs/year for 3 years, starting October 1, 2001

Distributed Computing Department
Lawrence Berkeley National Laboratory

Summary

High-performance distributed computing relies on high-speed network connectivity, which is in turn dependent on the availability of high-bandwidth physical links and computers fast enough to process the data packets. In recent years, the speed of CPUs and networks has increased consistently so that now, on scientific research networks, the theoretical maximum bandwidth available to applications is over one gigabit per second. This dramatic increase in network speed has not been realized because most applications' network utilization (fraction of the available bandwidth used by the application) is quite poor. This discrepancy between the experienced bandwidth and the physical bandwidth has been referred to as the "Wizard Gap". Although there are software factors limiting the bandwidth of connections, notably TCP settings such as buffer and congestion windows, one has to also take into account the current routing configuration and current bandwidth loads on shared network segments. The problem is that there are few tools that can provide accurate information about all these variables for high-speed networks, thus leaving one wondering where the bottleneck is.

To fully achieve the maximum bandwidth on a network is not an easy task. Troubleshooting the network elements and diagnosing the myriad of reasons for under-utilized bandwidth can be a daunting task. Network troubleshooting requires not only networking expertise, but also a set of very specialized software tools. Network wizards are aware that network elements can cause performance problem, but they need tools to help them identify problematic network elements. For example, if one could access every router in a network path and examine all the configuration information and statistics for each port (interface), then maybe one would be able to determine the status of a network path and all possible problems. Unfortunately, accessing every router on an arbitrary path through the Internet is a nearly impossible task, and reading each of the different routers' statistics requires extensive knowledge of a wide variety of routers. Another possible solution is to use the Simple Network Management Protocol (SNMP), a network management mechanism for monitoring and analyzing the status of network elements. Using SNMP, one can obtain detailed network statistics without having to logon to any network elements, such as routers, but SNMP requires "administrator" privileges, which most service providers managing the different routers along a path do not allow. In addition, SNMP programming has a steep learning curve, and there are not many experts in using it. In order to make network analysis and troubleshooting easier, tools that can be operated by average users and run at the user level without special privileges are needed. Using such tools, users can quickly identify problematic network elements and determine the best way to utilize the available network bandwidth.

Another task is to clarify what network characteristics are required by different users. Recently there has been a flurry of activity to develop network measurement tools. However, most of these tools either do not provide accurate results, especially on high-speed networks, or they only measure throughput, not available bandwidth. Users want to know both available bandwidth and achievable throughput. However, is bandwidth the only thing useful? What are other factors are useful? This is a critical issue to improving network performance. A key concept is missing during this flurry of development — feasibility and the factors that limits the achievement. Fortunately, people start to pay attention on this issue now, and they ask to have this issue related papers to be accepted by a number of technical conferences.

The goal of this project is to build a comprehensive tool that measures a number of useful characteristics, including available bandwidth, achievable throughput, maximum burst size, system capability, bottleneck, and so on. To do it, we create a network model and develop new algorithms based on this model to measure available bandwidth accurately in a non-intrusive way. The algorithms are not only suitable for measuring network bandwidth, but also may be used to help design a new network transmission protocol that can avoid packet loss and utilize the available bandwidth in order to maximize the network applications throughput on high-speed networks.

Project Goals and Objectives

Bandwidth is an important network characteristic, but there are often factors that can adversely affect network performance that are not network characteristics, but characteristics of the end hosts, such as memory bandwidth, I/O (input/output) bandwidth, the operating systems and applications (software). However, current research and development is focused only on the bandwidth measurement, and overlooks other possible factors. Because network bandwidth has doubled every year, network bandwidth is no longer the bottleneck that hinders the network applications' performance, especially over the high bandwidth delay product paths. A key issue to maximize the network performance is to have a set of information that can be used to develop new network transmission protocol to innovate or to replace the transmission control protocol (TCP), and help application developers understand what could be the bottleneck in end host systems that affect throughput, so the designer will make their software avoid the bottleneck in order to increase throughput.

Based on users experience on current network measurement tools, providing results only on network bandwidth is not enough information for users and applications to improve their throughput. This is because the available bandwidth and achievable throughput are not directly related, especially on high-speed networks. Several issues get mixed together that confuse bandwidth and throughput. That is, a high bandwidth network is an essential condition, but not the only condition for high throughput. For example, on a full 1 Gb/s path with 100 ms round trip delay, when available bandwidth is 10% (100 Mb/s) and most traffic is TCP, a new UDP stream running long enough will gain more than 90% (900Mb/s) total bandwidth (1 Gb/s). However, if the path is empty (1 Gb/s is available), an un-tuned TCP stream may only achieve 100 Mb/s throughput. Therefore, besides giving current available network bandwidth, a network measurement tool needs to tell how to maximize throughput to utilize this available bandwidth. If the current system configuration cannot achieve this goal, the tool needs explain why and how to improve it. This is the main goal of this project.

Additionally, this project is expecting to discover related network issues during the bandwidth algorithm research and development.

Technical Challenges

Creating a correct network model and developing algorithms for accurately measuring bandwidth is difficult. Educating people to understand that different system characteristics can be more critical to the performance than the available network bandwidth is even more difficult. Without a full understanding what and where the problem is, and people will not be able to find a solution for improving the performance.

The first task in this project is to develop algorithms to measure network bandwidth, especially available bandwidth, accurately and in a non intrusive way. This is a tacitly difficult problem.

The next challenge is how to implement the algorithms properly so as to build a measurement tool. Also, for such a measurement tool to be useful, it must be deployed and operated in a production environment. This means that the installation and operation must be as easy as possible, and the tool should be available to all commonly used hardware and UNIX operating systems. That is, it needs to be run on generic hardware without specific modification. Ideally, using the measurement tool does not require any network expertise, and its results should be easy to interpret. It should just download and run.

System Design Issues

System related issues occur at both hardware and software levels. Figure 1 shows a data path on a generic hardware system. The main bottleneck in current systems is at the memory and I/O sub system. Figure 1 shows the data path for sending data from user memory to the NIC. For a system equipped with a 64-bit/66MHz PCI bus, if the memory bus is 266 MHz, the total time needed to transfer data from a user buffer to the NIC is 6 memory cycles: the two fixed cycles plus four memory cycles per bus cycle (266/66). However if the memory bus is 533 MHz, then 10 cycles are required (2+533/66). The generic formula for calculating the I/O throughput from memory and I/O buses frequency is:

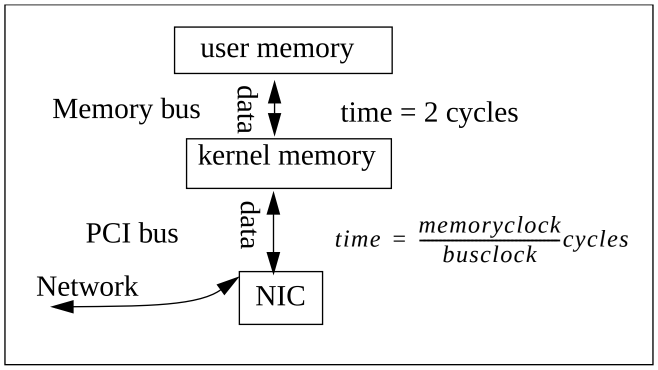

Figure 1: Relationship between I/O bandwidth and system memory bandwidth

$$IOthroughput = \frac{MemoryBandwidth}{(PCI + Memory \times 2)cycles} = \frac{MemoryBandwidth}{\frac{MemoryClock}{IOBusClock} + 2} \qquad (1)$$

In a real example, VIA 868 PCI controller has 133 MHz memory bus, and it produces 144 MB/s memory copy bandwidth (288 MB/s memory bandwidth). The newer generation VIA PCI controller (VT400) has 400 MHz memory bus, and produces 326 MB/s memory copy bandwidth (652 MB/s memory bandwidth). Both systems have 32-bit/33MHz PCI bus. Using equation (1), the VIA 868 based system can have maximum 384 Mb/s (48MB/s) network throughput, where VT400 based motherboard can only have 369 Mb/s of network throughput even though the new PCI controller has much higher memory bandwidth than the old one. In fact, VT400 based motherboard only had 300 Mb/s network throughput, where the older motherboard can achieve higher throughput by using the same NICs (network interface cards). Therefore, we can see that simply increasing the memory clock speed does not necessarily result in an equivalent increase in the data transfer rate from user space to the NIC.

Another issue is system calls and getting timestamp information via system resources. Table 1 lists the time to perform a system call on various O.S. platforms running on various CPUs. The time to perform a system call affects the outgoing packet pacing and the accuracy of getting timestamps for incoming packets. This is because the getting timestamp cost is at reading the CTC (TSC) rather than at the system call API — gettimeofday [10]. Therefore, timestamping packets inside the kernel does not provide better accurate than doing it at user level.

Table 1: Syscall time for some O.S.

	gettimeofday	read/write
Solaris 2.8 333MHz Sparc	348 ns	8400 ns
Solaris 2.7 400MHz Sparc	278-295 ns	5300 ns
AIX RS 6000	> 3000 ns	8500 ns
IRIX 2.6 175 MHz IP28	7946 ns	28162 ns
BSD/OS 526 MHz PII	10877 ns	11357 ns
Mac OS X 1GHz G4	1937 ns	2043 ns

Many high-speed NICs, including the SysKonnect card, provide an interrupt moderation feature, also known as interrupt coalescencing (or delayed interrupt), which bundles several packets into a single interrupt. The idea is that the NIC, on receipt of a packet, does not automatically generate an interrupt request to the CPU to process the data and release buffers for the NIC to get more packets. Instead, the interrupt is delayed for up to a given amount of time (the interrupt moderation period) in hopes of other packets arriving during that time can be served by the same interrupt.

Table 2 shows how interrupt coalescing affects CPU utilization, thus increasing the network throughput. TCP/IP packets are 1500 bytes for all measurements. From Table 2 we see that the receive host needed 92% of the CPU with default interrupt delay settings for the network interface card (NIC). After increasing the interrupt delay from 64 to 300 µs, the CPU usage is dropped to 72% due to the fact that fewer interrupts are generated. This means that the CPU has more time to processes packets, so throughput increased 85.9%.

Table 2: CPU utilization affected by I/O interrupt

interrupt delay time (coalescing)	% CPU IDLE	% CPU Interrupt	Throughput Mb/s
64 µs interrupt delay for Intel copper GigE (PCI/66) + Intel P4 Xeon 3 GHz CPU	0	92	277
300 µs interrupt delay for above configuration	1	72	515

If these system issues are not considered in engineering design for implementing measurement algorithms, they will introduce errors into the measurement results regardless of how accurate the algorithms are.

Milestones and Deliverables

Year 1: (10/01 - 10/02)

- Research and build a suitable network model for exploring network measurement algorithms
- Define what useful network characteristics that the tool(s) will measure
- Investigate and test hardware characteristics that affect measurement algorithm implementation
- Investigate characteristics of different operating systems on different hardware platforms
- Design algorithms for detecting and measuring system capabilities
- Do architecture design for building measurement infrastructure that will work on multi-platforms
- Do engineering design for implementing measurement of achievable throughput, optional number of parallel TCP streams [6]
- Design and implement throughput measurement algorithms

Year 2: (10/02-10/03)

- Design algorithms to measure available bandwidth and physical bandwidth based on the new network model
- Design algorithm for estimating number of parallel TCP streams instead of directly measuring it
- Investigate and test up-to-date hardware and provide results to public users
- Do engineer design for implementing bandwidth measurement
- Implement system capability detection and measurement
- Implement bandwidth measurement for end-to-end measurement
- Test netest over different network and verify the accurate via emulation network and ESnet
- Disseminate the achievement and new tools for people to use
- Incorporate new tools into other measurement infrastructures.

Year 3: (10/03-10/04)

- Experiment FAC^2 (feedback adaptive control and feedback asymptotic convergence) algorithm for hop-by-hop measurement
- Test algorithms and tool on more different network paths
- Continue to disseminate the high efficient bandwidth measurement tool
- Continue to investigate and test up-to-date hardware and provide information to the public
- Research on integrate FAC^2 algorithm and maximum burst size (MBS) theory into network protocol for building a better protocol in utilizing the high delay bandwidth product network.

Note: This project is incorporated with the Net100 project, and is used by Net100 to measure bandwidth, achievable throughput and to detect the bottleneck along end-to-end paths.

Project Progress To Date

Currently this project has completed the network model, mathematical algorithms, hardware and software architecture, and software engineering designs; completed implementation of the comprehensive measurement software for measuring following network characteristics for end-to-end paths:

- bandwidth (available and physical)
- achievable throughput — the best throughput applications can reach
- system limitation — where is the bottleneck along the end-to-end path beyond the network
- maximum burst size — the critical factor affecting the network bandwidth utilization — maximum packets transferred in one burst without causing loss
- round trip delay
- number of parallel TCP streams for improving TCP performance over long delay bandwidth product paths
- network status
- recommendation for application performance improvement

The tool is called — netest (revision 2) is available for multiple platforms.

This project has some significant contributions to the future network development and measurement:

(1) Fluid Spray Effect: this is the theory used for detecting congestion and passing through the congestion point, where packets can be separated in distance, to measure bandwidth further.

(2) Maximum burst size (MBS) theory: this is an important concept to avoid packet drop in the network. It is critical to both bandwidth measurement and network transmission protocol. The *maximum burst size* is determined by the queue size of the bottleneck router and by the current network traffic at that router. Note that other factors such as traffic shaping, policing, or QoS may cause such queueing behavior. This can restrict the effective bandwidth when it is small. So, MBS is also called as *effective queue size*. The following example depicts how MBS significantly affects throughput.

This example illustrates that if MBS is smaller than BDP (bandwidth delay product), it will reduce the effective path bandwidth for normal operation. In this example, the maximum throughput was reduced to one half of the capacity. If applications or operating systems are not aware of this issue, they cannot utilize the available bandwidth and will cause congestion which degrades network performance. MBS is also critical to the packet train method for network measurement. That is, the maximum train length must be less than the MBS; otherwise, the train's tail will be dropped, causing probe failure. How long the train should be configured depends on the variation of the cross traffic.

This example uses a path with 100 ms round trip delay (RTT), while the bottleneck link is 1 Gb/s. In order to maximize TCP throughput, TCP needs to set congestion window to the bandwidth delay product (BDP)

$$0.1s \times 10^9 b/s = 100Mbits = 12.5MBytes$$

If the MBS is 1 MB due to the average cross traffic, the TCP congestion window will be limited to 1 MB because any burst larger than 1 MB will cause packet loss when cross traffic exists, which is almost always true. The maximum TCP throughput then will be

$$\frac{1MB \times 8bits/Byte}{0.1s} = 80Mb/s$$

UDP throughput also depends on the burst size and rate of the traffic. 1 MB effective queue implies that the maximum burst duration (at line speed, 1 Gb/s in this case) for both probe traffic and cross traffic to avoid overflow is:

$$\frac{MBS}{LineSpeed} = \frac{1MB}{1Gb/s} = 8ms$$

because when a burst leaves the previous router, it will travel at link (line) speed. Since we can characterize all cross traffic as one aggregated stream, each stream (aggregated cross traffic and measurement traffic) has minimum 8 ms safety burst period (for router to drain two 1 MB bursts in average according to the effective queue size) to not overflow the smallest queue on this path. Without knowing how to control the burst, UDP throughput will vary depending on how many cross traffic bursts have a duration longer than 8 ms (exceed 1MB in length). The more big bursts of cross traffic, the lower the UDP throughput will be.

(3) FAC^2 (feedback adaptive control and feedback asymptotic convergence): this is the algorithm that can accurately measure available bandwidth in five (5) round trip time with very low intrusiveness. It can also analyze the cross traffic, thus to estimate bottleneck router's physical bandwidth and average queue size. This algorithm has been mathematically proven to be accurate and LBNL has filed patent protection on it. This algorithm is also critical to building a reliable network transmission protocol.

Results:

Since there is no enough space to do mathematical proof [LBNL-53165] on all theories and algorithms used in *netest* — a comprehensive tool for measuring important characteristics between end hosts including network bandwidth, Table 3 provides physical proof on how accurately *netest* can measure available bandwidth on a 1 Gb/s emulation network provided by CAIDA[11]. The emulation network topology can be found at [15]. Netest is able to measure available bandwidth up to 4 Gb/s based on current on-shelf hardware from market.

Table 3: available bandwidth results

Utilization % (loss %)	`netest` Mb/s	Accuracy (%)
GigE network MTU = 9K 50~100 tests	**typical measurement duration** (*require longer duration*) **2.4 - 6.5 seconds** <including MBS measurement>	
0 (0)	maximum throughput: 851 [10]	
10 (0)		
20 (0)	791.0 - 791.2	98.875
30 (0)	690.0- 691.0	98.643
40 (0)	598.5 - 599.0	99.750
50 (0)	502.5 - 502.9	99.420
60 (0)	403.8 - 403.9	99.025
70 (0)	306.4 - 306.6	97.833
80 (0.01)	210-211 (*7.89 sec.*) 205 (*11.9 sec.*)	97.500
90 (0.01)	113-115 (*15 sec.*) 102 (*26 sec.*)	98.000

Related Work

Several active network bandwidth and throughput measurement tools are currently available. Often tools for bandwidth measurement are pathchar [4][14] and pathload [7]. A number of tools is good for measuring the throughput, such as netest version 1, netperf [13] and iperf [12].

Pathchar is a tool measuring physical bandwidth (capacity). It has an outstanding algorithm that can use a slow NIC to measure a high-speed network capacity. The issues of this algorithm are: (1) it needs long time to get result; (2) due to the maximum data size is limited to the one MTU (maximum transfer unit, typically 1500 bytes), its accuracy is only valid for link speed of 200 Mb/s or less.

Pathload is a tool to measure available bandwidth. Its algorithm theoretically can measure available bandwidth fairly accurately if bisection method is applied properly. The current implementation lacks of engineering design on hardware and operating system related issues, so it only works for NICs that are OC-12 and slower. All Gigabit NICs have interrupt moderation which pathload cannot currently handle.

Iperf is a common tool to measure throughput because it is flexible and easy to use. Netperf is another similar tool.

Netest version 1 is the first tool that uses MBS and non intrusive mechanisms to measure throughput. It is also one of the first tool to do diagnosing of network problems, such as how packets are dropped.

Publications

- **G. Jin, G. Yang, B. Crowley, D. Agarwal, Network Characterization Service (NCS), HPDC-10 Symposium, August 2001, LBNL-47892**
- **Guojun Jin, Algorithms and Requirements for Measuring Network Bandwidth, LBNL technical report, Jan. 2003, LBNL-48330**
- **G Jin, B Tierney, Netest: A Tools o Measure the Maximum Burst Size, Available Bandwidth and Achievable Throughput. ITRE, Aug. 2003, LBNL-48530**
- **D. Agarwal, J. M. González, G. Jin, B. Tierney An Infrastructure for Passive Network Monitoring of Application Data Streams, 2003 Passive and Active Measurement Workshop, LBNL-51846**
- **Guojun Jin, Feedback Adaptive Control and Feedback Asymptotic Convergence Algorithms for Measuring Network Bandwidth, Submitted to InfoCOM 2004, July 2003, LBNL-53165**
- **G Jin, B Tierney, System Capability Effects on Algorithms for Network Bandwidth Measurement, IMC, Oct. 2003, LBNL-48556**

Future Work

The tasks left for the 3rd year of the project are:

- Disseminate netest to the world so that users can start to use it to understand the issues affecting network performance.
- Experiment hop-by-hop measurement via FAC^2 algorithm.
- Show how FAC^2 and MBS is critical to the next generation network transmission protocol and how to build a new network transmission protocol.
- Find partner(s) who want to do collaboration on developing and building new network transmission protocol.

Interactions

This project is working closely with the Net100 project in determining what characteristics applications and developers need to know; in developing tools to measure those characteristics; in testing tools; and in determining and testing the hardware requirement for building testbeds. The hardware information is also used for the Bro intrusion detection system and SCNM development. We are working closely with SCNM, ESNet and ORNL networking engineers to test netest and to track down networking problems using netest. We have been working closely with Jiri Navratil and Les Cottrell at SLAC for intensively testing netest in PingER, and working closely with Margaret Murray, Kimbley Claffy and Grant DuVall at CAIDA for evaluating measurement tools and verify their accuracy. We have also been working with Constantine Dovrolis and Manish Jain at Georgia Tech doing bandwidth measurement development.

References

[1] S. McCanne and V. Jacobson, “The BSD Packet Filter: A New Architecture for User-level Packet Capture,” Proc. Winter USENIX Technical Conference, Jan. 1993.

[2] V. Paxson, “Measurements and Analysis of End-to-End Internet Dynamics”, Ph.D. dissertation, 1997.

[3] Kevin Lai and Mary Baker. Measuring Bandwidth. In Proceedings of IEEE INFOCOM, March 1999.

[4] Allen B. Downey, Using pathchar to estimate Internet link characteristics, proceedings of SIGCOMM 1999, Cambridge, MA, September 1999, 241-250.

[5] Kevin Lai and Mary Baker, "Nettimer: A Tool for Measuring Bottleneck Link Bandwidth", Proceedings of the USENIX Symposium on Internet Technologies and Systems, March 2001.

[6] Thomas J. Hacker, Brian D. Athey, The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network, Aug. 2001.

[7] Manish Jain and C. Dovrolis, Pathload: A Measurement Tool for End-to-end Available Bandwidth, PAM, March, 2002.

[8] Attila Pasztor´, Darryl Veitch, PC Based Precision Timing Without GPS, Sigmetrics, June 2002

[9] Attila Pásztor, Darryl Veitch, Active Probing using Packet Quartets, IMW, Nov. 2002

[10] G Jin, B Tierney, System Capability Effects on Algorithms for Network Bandwidth Measurement, IMC, Oct. 2003

[11] caida: http://www.caida.org/

[12] iperf: http://dast.nlanr.net/Projects/Iperf

[13] Netperf: A Network Performance Benchmark. Available: http://www.netperf.org/netperf/training/Netperf.html

[14] pchar: A Tool for Measuring Internet Path Characteristics. Available: http://www.employees.org/~bmah/Software/pchar

[15] http://www.didc.lbl.gov/NCS

[16] Distributed Monitoring Framework: http://www-didc.lbl.gov/DMF/